A PUBLIC KEY CRYPTOGRAPHIC METHOD OF PROTECTING AN ELECTRONIC CHIP AGAINST FRAUD

FIELD OF THE INVENTION

The field of the present invention is that of cryptography. The invention relates in particular to cryptographic methods of protecting an electronic chip against fraud in transactions between the chip and an application.

The invention finds a highly advantageous application in protecting hard-wired logic or microprocessor-based integrated circuit chips from fraud, in particular the chips in prepaid cards used in diverse transactions, such as making telephone calls, purchasing items from an automated dispenser, paying parking charges at a parking meter, and paying for a service such as a public transport service or for the provision of infrastructures (tolls, museums, libraries, etc.).

BACKGROUND OF THE INVENTION

At present, prepaid cards are open to various types of fraud. A first type of fraud consists in unauthorized duplication of the card, often known as "cloning". A second type of fraud consists in modifying the data associated with a card, in particular the amount of credit registered in the card. Cryptography is used to combat these kinds of fraud, firstly by authenticating the card and/or data by means of a digital signature, and secondly by using encryption when it is necessary to protect the confidentiality of the data. Cryptography, which can be either symmetrical or asymmetrical, uses two entities, which in the case of authentication comprise a verifier and an object to be verified. When cryptography is symmetrical (or of the "secret key" type, these two terms being interchangeable), the two entities share exactly the same information, in particular a secret key. When cryptography is asymmetrical (or of the "public key"

type, these two terms being interchangeable), one of the two entities has a pair of keys of which one is secret and the other is public; there is no shared secret key. Many systems use only symmetrical cryptography for prepaid cards, especially when the chip is of the "hard-wired logic" type, because asymmetrical cryptography is still slow and costly. The first authentication mechanisms developed for symmetrical cryptography comprises calculating once and for all an authentication value that is different for each card, storing it in the memory of the card, reading it during each transaction, and verifying it by interrogating an application of the network supporting the transaction and in which authentication values that have already been assigned are either stored or recalculated. Those mechanisms provide insufficient protection, since the authentication value can be misappropriated, reproduced, and played back fraudulently, because it is always the same for a given card, enabling the card to be cloned. To combat cloning, passive card authentication mechanisms are replaced by active authentication mechanisms that can also assure data integrity.

The general principle of active symmetrical authentication mechanisms is as follows: during authentication, the electronic chip and the application calculate an authentication value by applying a function to a list of arguments determined at the time of each authentication; the list of arguments can include, firstly, a random number, which is an item of data determined by the application at the time of each authentication, secondly, an item of data contained in the electronic chip, and, thirdly, a secret key known to the electronic chip and to the application. If the authentication value calculated by the electronic chip is identical to the authentication value calculated by the application, the electronic chip is deemed to be authentic and the transaction between the electronic chip

and the application is authorized.

Authentication mechanisms of the above kind are well known in the art, but most of them demand calculation capacities at least equal to those of a microprocessor. Those mechanisms are therefore suitable for microprocessor-based cards, but are rarely suitable for hard-wired logic chips, which have calculation capabilities that are much more rudimentary.

A first step forward was achieved when it became possible to integrate active symmetrical authentication mechanisms into hard-wired logic chips. For example, French Patent Application No. FR 2 826 531 published on December 27, 2002 describes a method of specifying such mechanisms. It should be observed that, as taught by the above-mentioned French Patent Application, the authentication value produced by those mechanisms can also be interpreted as a sequence of pseudo-random bits and, by varying at least one of the input parameters, the method of calculating the authentication value becomes a method of generating pseudo-random bits.

However, secret key mechanisms require the verification device responsible for authenticating the chip, such as a device in a public telephone, an electronic payment terminal, or a public transport gate, to know the secret key held by said chip. This is a major drawback in that, if said device is required to be able to authenticate any chip issued in relation to the application, it must store either the secret keys of all the chips or a basic key, or master key, or mother key as it is otherwise known, enabling it to determine the secret key of any chip. In both cases, each device stores sufficient information to be able to determine the secret keys of all the chips issued, and therefore stores sufficient information for cloning any of them. It follows that successful hacking into any of the verification devices would negate the security of the entire application.

OBJECTS AND SUMMARY OF THE INVENTION

One object of the present invention is to integrate an active public key authentication mechanism into a hard-wired logic chip, in particular in applications deploying a large number of chips, which is generally the case with applications using hard-wired logic chips because they are of very low cost. No such mechanism exists at present. The reason for this is that public key mechanisms generally require numerous operations on large numbers, and are therefore unsuited to integration in hard-wired logic chips, in which the surface area of the silicon is extremely small, and whose calculation logic is reduced to extremely basic hard-wired operations. These basic operations are generally effected serially, in the sense that the operands are introduced sequentially, bit by bit, and this progressively modifies the state of an internal register whose final value serves as a basis for calculating the result of the function.

The present invention relates to active public key authentication mechanisms that can be implemented in a hard-wired logic card.

To be more precise, the present invention relates to an asymmetrical cryptographic method of protecting an electronic chip against fraud in transactions between the electronic chip and an application, more particularly suitable for hard-wired logic chips and more particularly intended for implementing an authentication mechanism that is free of the symmetrical cryptography drawbacks mentioned previously, so as to enhance the security of the entire application, and in particular so as to make cloning more difficult.

The above-mentioned object and other objects are attained in accordance with one aspect of the present invention directed to an asymmetrical cryptographic method of protecting an electronic chip against fraud in transactions between the electronic chip and an

application, involving calculating an authentication value V from input parameters in the electronic chip. The chip produces a pseudo-random number r specific to the transaction by means of a serial pseudo-random generator included in the chip. The chip sends the application a parameter x calculated by the application prior to the transaction, linked to the random number r by a mathematical relationship, and stored in a data memory of the chip. The chip calculates a parameter y constituting the whole or a portion of the authentication value V by means of a serial function whose input parameters are at least the random number r specific to the transaction and a private key s belonging to an asymmetrical pair of keys. The chip sends the authentication value V to the application, and the application verifies the authentication value V by means of a verification function whose input parameters consist exclusively of public parameters including at least the public key p.

Another aspect of the invention is directed to an electronic chip device adapted to implement the above asymmetrical cryptographic method of protecting the electronic chip against fraud in transactions between the electronic chip and an application, by the electronic chip calculating an authentication value V from input parameters. The device comprises a serial pseudo-random generator for producing a random number r specific to the transaction, first memory means for storing one or more values of the parameter x calculated prior to the transaction by the application and linked by a mathematical relationship to the value of the random number r, means for sending the parameter x linked to the random number r specific to the transaction from the chip to the application, means for executing a serial function having as input parameters at least the random number r specific to the transaction and a private key s belonging to an asymmetrical pair of keys (s, p) and providing as

output a parameter y, and output means adapted to construct the authentication value V from at least the parameter y.

Another aspect of the invention is directed to a verification device for executing an asymmetrical cryptographic method of the invention for protecting an electronic chip against fraud in transactions between the electronic chip and an application. Such method includes verifying an authentication value V calculated by the electronic chip from exclusively public parameters. The device comprises means for executing the verification function taking as input at least the authentication value V and the public key p.

A method according to the invention has the advantage of enabling an authentication value V to be produced that can be verified by means of public parameters alone, although it is produced exclusively by serial functions, i.e. functions that process sequentially the bits of the parameters that constitute its input.

The input parameters of the cryptographic method and device are processed in the serial function that supplies as output an item of data dependent on some or all of the input parameters.

The input parameters of the method and device belong to a list that, in the case of implementing an authentication mechanism, comprises at least an identifier I, a private secret key s, a public key p corresponding to the private key s, a certificate for said public key, and a second random number t provided by the verification device.

The serial pseudo-random generator for calculating the random number r can advantageously be based on a symmetrical authentication method of the type described in the above-mentioned French Patent Application No. FR 2 826 531 published on December 27, 2002. Accordingly, if f(K, M) designates the calculation

function of a method of this kind, where K designates the symmetrical secret key and M designates all other operands of the function f, then the random number r can be produced by repeated application of the function f to different values of M, retaining the same value of K. For example, if the size of the output value z of f is equal to k bits and if the size of the random number r is equal to 16k bits, the first random number r used in the first authentication of the chip can be made equal to the concatenation of the sixteen output values $f(K,M_1)$, $f(K,M_2)$, ..., $f(K,M_{16})$; the second random number can be made equal to the concatenation of the sixteen output values $f(K,M_{17})$, $f(K,M_{18})$, ..., $f(K,M_{32})$, etc., all the values M_i being different from each other (the value of M_{i+1} is typically obtained by incrementing the value of M_i). There are many other ways to use the authentication method for pseudo-random generation.

The serial function contains additions, subtractions and left- or right-shifts. These operations can very easily be performed sequentially.

BRIEF DESCRIPTION OF THE DRAWINGS

Other features and advantages of the invention become apparent in the course of the following description given with reference to the accompanying drawings of particular embodiments provided by way of non-limiting example.

Figure 1 is a flowchart of a method in accordance with the invention.

Figure 2 is a diagram of an electronic chip device in accordance with the invention.

Figure 3 is a diagram of one embodiment of a pseudo-random generator for an electronic chip device in accordance with the invention.

Figure 4 is a diagram of one embodiment of means for executing a serial function for an electronic chip device in accordance with the invention.

DETAILED DESCRIPTION OF THE DRAWINGS

Figure 1 is a flowchart of an asymmetrical cryptographic method of the invention for protecting an electronic chip against fraud in transactions between the electronic chip and an application.

The method includes performing a calculation in the chip to determine an authentication value from input parameters.

In a first step 1 of the method, the chip produces a pseudo-random number $\underline{r}$ by means of a serial pseudo-random generator included in the chip. The random number $\underline{r}$ is specific to the transaction.

In a second step 2 of the method, a parameter $\underline{x}$ is sent from the chip to the application. The parameter $\underline{x}$ is calculated prior to the transaction by the application and is stored in the data memory of the chip. The parameter $\underline{x}$ is linked to the random number $\underline{r}$ by a mathematical relationship. The application calculates at least one parameter $\underline{x}$ and preferably calculates several parameters. In one particular embodiment, the parameters $\underline{x}$ are the result of applying a mathematical function to values taken successively from a given set for a given chip. The set is such that the various values of the random number $\underline{r}$ generated by the chip are included in that set.

Thus the mathematical function linking the random numbers $\underline{r}$ and the parameters $\underline{x}$ typically comprises an exponential function in a set G provided with an operation having at least the property of being associative and denoted in the form of a multiplication, i.e. the function is $x = g^r$, where $\underline{r}$ designates an integer and $\underline{g}$ designates an item from said set G chosen beforehand by the application.

The pseudo-random number $\underline{r}$ is different for each chip and for each authentication. It is calculated twice, the first time by the application and the second time by the chip itself. After calculating $\underline{r}$, the

application calculates the corresponding $\underline{x}$. The application then stores at least one value of $\underline{x}$ in the chip when customizing it. The application advantageously stores a plurality of values of $\underline{x}$. Since the application and the chip must produce the same value of $\underline{r}$, it is of course imperative for the pseudo-random generator of the application and that of the chip to be strictly identical.

$\underline{g}$ can advantageously be either the same for all the electronic chips linked to the application or specific to the chip. When specific, $\underline{g}$ is an integral part of the public key $\underline{p}$ of the electronic chip. Typical examples of sets G are the group Z_n^* of positive or zero integers less than $\underline{n}$ and prime with $\underline{n}$ (where $\underline{n}$ designates any positive integer), or any elliptical curve constructed on a finite body.

In a third step 3 of the method the chip calculates a parameter $\underline{y}$ by means of a serial function whose input parameters are at least the random number $\underline{r}$ specific to the transaction and a private secret key $\underline{s}$ belonging to an asymmetrical pair of keys (s, p), the parameter $\underline{y}$ constituting the whole or a portion of the authentication value V. The serial function is an arithmetical function.

In a fourth step 4, the chip sends the authentication value V to the application.

In a fifth step 5 of the method, the application verifies said authentication value V by means of a verification function whose input parameters are exclusively public parameters and include at least the public key $\underline{p}$.

Figure 2 shows diagrammatically a device of the invention including an electronic chip. The device executes an asymmetrical cryptographic method of the invention for protecting the electronic chip against fraud in transactions between the electronic chip and an application, the method comprising the electronic chip

calculating an authentication value V from input parameters.

The device 6 comprises:

- a serial pseudo-random generator 7 producing a random number $\underline{r}$ specific to the transaction,

- first memory means 8 for storing one or more parameters $\underline{x}$ calculated by the application prior to the transaction, each of said parameters $\underline{x}$ being linked by the same mathematical relationship to a value of the random number $\underline{r}$ within a set of values that can be produced by the serial pseudo-random generator,

- first output means 9 for supplying the parameter $\underline{x}$ linked to the random number $\underline{r}$ specific to the transaction,

- means 10 for executing a serial function having for input parameters at least the random number $\underline{r}$ specific to the transaction and a private key $\underline{s}$ belonging to an asymmetrical pair of keys (s, p), said parameter $\underline{y}$ constituting the whole or a portion of the authentication value V, and

- second output means 9 for supplying the authentication value V after said value is constructed from at least the parameter $\underline{y}$.

In the embodiment described with reference to Figure 2, the serial pseudo-random generator 7 uses a symmetrical authentication method of the type described in the above-mentioned French Patent Application No. FR 2 826 531 published on December 27, 2002. Accordingly, if f(K,M) designates the calculation function of a method of this kind, in which K designates the symmetrical secret key and M designates all other operands of the function $\underline{f}$, then the random number $\underline{r}$ can be produced by repeated application of the function $\underline{f}$ to different values of M, retaining the same value of K. For example, if the size of the output value $\underline{z}$ of $\underline{f}$ is equal to $\underline{k}$ bits and if the size of the random number $\underline{r}$ is equal to 16k bits, the first random number $\underline{r}$ used in the

first authentication of the chip can be made equal to the concatenation of the sixteen output values $f(K,M_1)$, $f(K,M_2)$, ..., $f(K,M_{16})$; the second random number can be made equal to the concatenation of the sixteen output values $f(K,M_{17})$, $f(K,M_{18})$, ..., $f(K,M_{32})$, etc., all the values M_i being different from each other.

Figure 3 is a diagram of a serial pseudo-random generator 6 of the above kind. The generator comprises, firstly, means 12 for mixing some or all of the input parameters to supply at its output a data item $E' = (e'_1, e'_2, ..., e'_n, ..., e'_N)$ resulting from said mixing, secondly, a finite state automaton 13 that changes from an old state to a new state in accordance with a function depending at least on the old state and a value from the series of bits $(e'_1, e'_2, ..., e'_n, ..., e'_N)$, and, thirdly, output means 14 for calculating the value $\underline{z}$ from input arguments comprising at least one state of the automaton, and thereafter determining the value of the random number $\underline{r}$ chosen by concatenating sixteen successive output values $f(K,M_1)$, $f(K,M_2)$, ..., $f(K,M_{16})$. The input parameters of the mixing means 12 can be: a secret key K, data D internal to the chip, the memory address of the data D, data D' external to the chip, and a random number R' supplied by the application (this list is by no means exhaustive).

The mixing means 12 execute a mixing function MIX that can be a linear or non-linear function of the input data.

The scalar product of the input data is a first example of a linear function.

In another example of a linear function, the mixing means comprise a linear feedback shift register into which the bits of the input parameters are entered successively and influence the initial state of the register and/or the value of the feedback bits.

In a further example of a non-linear function, the mixing means comprise a non-linear feedback shift

register into which the bits of the input parameters are entered successively. The output value S' can comprise one or more bits extracted from the content of this register.

A first example of the automaton 13 uses a Boolean circuit, i.e. a circuit which associates a vector (A'_1, A'_2, ..., A'_k) of $\underline{k}$ bits with a vector (A_1, A_2, ..., A_{k+1}) of k+1 bits, where each bit A'_i is obtained from the bits (A_1, A_2, ..., A_{k+1}) using basic operations such as exclusive-OR, OR (inclusive), AND, and NOT operations, and where (A_1, A_2, ..., A_k) represents the old state of the automaton. The automaton has an internal state (A_1, A_2, ..., A_k) of $\underline{k}$ bits and outputs a new state (A'_1, A'_2, ..., A'_k) each time that a new vector (A_1, A_2, ..., A_k, S'e') is present at the input of the Boolean circuit, the new vector comprising the internal state and the output of the mixing function MIX.

A second example of the automaton 13 uses bit transforms defined by tables of numbers. If k = 8, it is possible to divide the byte (A_1, A_2, ..., A_8) into two quads (A_1, A_2, A_3, A_4) and (A_5, A_6, A_7, A_8), for example, and then to apply to each quad either a transform T if the value of the output bit E'e' of the mixing function is 0 or a transform U if the value of E'e' is 1. The transform T is defined by a table that associates a quad value (a', b', c', d') with each quad value (a, b, c, d), and the same applies to the transform U.

When all the input values have been processed, the automaton 13 is in a certain final state (F_1, F_2, ..., F_k).

The output means 14 of the serial pseudo-random generator typically use an output function that is the identity function applied to the final state of the automaton and a concatenation operation. The output means 14 comprise a memory region whose size is equal to the size of the random number $\underline{r}$, for example, which is 16k bits.

The first memory means 8 for storing one or more

parameters $\underline{x}$ typically comprise a non-volatile memory, possibly one that can be rewritten. The parameters $\underline{x}$ are programmed into the memory before the electronic chip is sold. The value of the random number $\underline{r}$ used to calculate the parameter $\underline{x}$ must be chosen so that the chip can calculate exactly the same value. In the serial pseudo-random generator described by way of example with reference to Figure 2, this implies that the secret key K is shared by the chip and the application. Accordingly, before the chip is put into circulation, the application calculates a number of values of $\underline{x}$ by repeatedly applying the authentication method whose calculation function is denoted $\underline{f}$ hereinabove, and stores these values in the data memory of the chip. On each authentication, the chip recalculates the random number $\underline{r}$ and reads the corresponding value of the parameter $\underline{x}$ in the data memory. In the serial pseudo-random generator described by way of example with reference to Figure 2, the correspondence between $\underline{r}$ and $\underline{x}$ is typically established by choosing for the value of M_1 information for determining the address of the value of $\underline{x}$ corresponding to that particular value of $\underline{r}$, the value of M_{i+1} for $\underline{i}$ greater than or equal to 0 being obtained by incrementing the value of M_i.

To economize on memory space, the parameter $\underline{x}$ can advantageously be made equal to the image of the item g^r (and possibly other items, such as application data) produced by a hashing function $\underline{h}$, rather than have it be equal to the item itself; in other words: $x = h(g^r, D)$, where D designates an optional field containing data related to the application, for example. D designates an amount in Euros decided on by the application, for example. In this case, each coupon represents an electronic coin and each authentication represents the spending of a coin.

The first output means 9 that output the parameter $\underline{x}$ linked to the random value $\underline{r}$ specific to the transaction

typically comprise an input/output buffer.

One example of the means 10 for executing a serial function is described next with reference to Figure 4. The input parameters of the serial function are the random number $\underline{r}$ and a private secret key $\underline{s}$ belonging to an asymmetric pair of keys (s, p). The key $\underline{p}$ is a public key.

The means 10 comprise a bit adder that calculates and takes into account a carry.

The value of the current bit r_i of $\underline{r}$ is captured in a first register 15 and the value of the current bit s_i of $\underline{s}$ is captured in a second register 16. A third register 17 captures the carry c_i from previous bit additions. Finally, a fourth register 18 captures the bit y_i obtained after addition of the values of the current bits r_i and s_i with the carry obtained in the preceding addition and corresponding to the content of the third register 17. The carry c_i results from taking account of the carry generated on adding the preceding bits (the output of the AND gate 19, whose inputs are the outputs of the first two registers 15, 16) and the carry generated on adding the current bits (the output of the AND gate 20 whose inputs are the values of the current bits r_i and s_i). An intermediate AND gate 21 generates a carry if a carry is generated on adding the preceding bits and when only one of the current bits is a 1, at the output of the exclusive OR gate 22 whose inputs are the values of the bits.

The carry is therefore the result of an OR operation 23 between the output of the intermediate AND gate 21 and the AND gate 20 whose inputs are the values of the current bits r_i and s_i. This carry c_i is captured in the third register 17 to be taken into account on adding the next bits of r_i and s_i.

The bit y_i results from adding the values of the current bits r_i and s_i (at the output of the exclusive-OR gate 22 whose inputs are the values of the current bits r_i

and s_i) and the value of the carry (at the output of the exclusive-OR gate 24 whose inputs are the output of the preceding exclusive-OR gate 22 and the output of the third register 17).

The outputs of the registers 15, 16, 17, 18 are initialized to 0.

This finally yields: $y_i = r_i + s_i + c_i(\text{mod}2)$ and $c_{i+1} = r_i + s_i + c_i(\text{div}2)$, where c_0 is made equal to 0.

In one particular application, the serial function has a further input parameter in the form of a random number $\underline{t}$ supplied by the application.

After the chip has produced a random number $\underline{r}$ by the method described with reference to Figure 2 and has then read the value of the parameter $\underline{x}$ in its data memory that corresponds to the value of said random number (for example using the function $x = g^r$), it sends the value of $\underline{x}$ to the application, whereupon the application sends the chip a random number $\underline{t}$ whose size is reduced to 1 bit.

Two situations then arise: if the value of $\underline{t}$ is 0 the chip chooses $y = r$ and if the value of $\underline{t}$ is 1 the chip chooses $y = r + s$. How to implement this choice is well known to the person skilled in the art and is therefore not described here.

The authentication value V is taken as equal to $\underline{y}$ and is sent to the application.

Verification comprises testing the equation $g^y = x$ if $\underline{t}$ is equal to 0 or $g^y = xp$ if $\underline{t}$ is equal to 1, where $\underline{p}$ is the public key of the chip corresponding to its secret key $\underline{s}$, as defined by the function $p = g^s$. If these parameters are made sufficiently large, it is not feasible to determine $\underline{s}$ from $\underline{g}$ and $\underline{p}$ using the discrete logarithm hypothesis, which is widely accepted at present.

In one particular application, a hashing function $\underline{h}$ can be used to calculate $\underline{x}$. In this case, the verification equation becomes $h(g^y, D) = x$ if $\underline{t}$ is equal to 0 or $h(g^y/p, D) = x$ if $\underline{t}$ is equal to 1. To avoid any

division in the verification equation, it is also possible to choose $y = r - s$ rather than $y = r + s$, in which case the second verification equation becomes $h(g^y.p,D) = x$. Another option is to choose $p = g^{-s}$ rather than $p = g^s$, which yields the following verification equations: $h(g^y,D) = x$ and $h(g^y.p,D) = x$.

In the embodiments previously described, any chip other than the one that knows the secret value s has at most one chance in two of supplying an authentication value that the application recognizes as valid. This already makes a distinction between an authentic chip and a clone, but this distinction is insufficient in most real-life situations.

To reduce significantly the chances of successful cloning, one solution is to increase the number of bits m of the random number t. For example, the random number t can be a string of 64 bits (t_{63}, t_{62}, …, t_0) in which only one bit is equal to 1. If i is the only suffix such that t_i is equal to 1, then the value of y is made equal to $y = r + 2^i s$, which is very easy to calculate sequentially, since this amounts to adding r and the integer obtained by left shifting s by i bits (if the more significant bits are on the left). The verification equation is then $g^y = xp^{2^i}$. Under these conditions, any chip other than the one knowing the secret value s has at most one chance in 64 of supplying an authentication value that the application would recognize as valid.

In one particular application, a hashing function h can be used to calculate x. In this case, the verification equation becomes: $h(g^y/p^{2^i},D) = x$. To avoid any division in the verification equation, it is also possible to choose $y = r - 2^i s$ rather than $y = r + 2^i s$, in which case the second verification equation becomes $h(g^y.p^{2^i},D) = x$. Another option is to choose $p = g^{-s}$ rather than $p = g^s$, which yields the verification equation $h(g^y.p^{2^i},D) = x$.

For the solution as described, and from the security point of view, it amounts to the same thing to choose for the value of $\underline{t}$ an integer from 0 to m - 1 instead of the string $\underline{t}$ defined above, in which case $\underline{y}$ is taken as equal to $y = r + 2^t s$ and the verification equation is $g^y = xp^{2^t}$.

In one particular application, a hashing function $\underline{h}$ can be used to calculate $\underline{x}$. In this case, the verification equation becomes: $h(g^y/p^{2^t}, D) = x$. To avoid any division in the verification equation, it is also possible to choose $y = r - 2^t s$ rather than $y = r + 2^t s$, in which case the second verification equation becomes $h(g^y.p^{2^t}, D) = x$. Another option is to choose $p = g^{-s}$ rather than $p = g^s$, which yields the verification equation $h(g^y.p^{2^t}, D) = x$.

From the security point of view, it amounts to the same thing to choose for the value of $\underline{t}$ an integer from 0 to m - 1 instead of the string $\underline{t}$ defined above, in which case $\underline{y}$ is taken as equal to $y = r + ts$ and the verification equation is $g^y = xp^t$.

In one particular application, a hashing function $\underline{h}$ can be used to calculate $\underline{x}$. In this case, the verification equation becomes: $h(g^y/p^t, D) = x$. To avoid any division in the verification equation, it is also possible to choose $y = r - ts$ rather than $y = r + ts$, in which case the second verification equation becomes $h(g^y.p^t, D) = x$. Another option is to choose $p = g^{-s}$ rather than $p = g^s$, which yields the verification equation $h(g^y.p^t, D) = x$.

The random number $\underline{t}$ can of course take other values.

The second output means 9 (authentication value V) typically use an output function that is the identity function applied to the parameter $\underline{y}$. The second output means 9 comprise a memory region whose size is equal to the size of the parameter $\underline{y}$, for example.

During transactions between an application and an electronic chip of the invention, the application and the

chip use an asymmetrical cryptographic method to protect the electronic chip against fraud, in which method the application uses a verification device of the invention to authenticate the chip. The device comprises means that execute the verification function of a method of the invention to verify the authentication value V calculated by the electronic chip using only public parameters that comprise at least the public key p linked to the secret key s of the chip.

In one of the embodiments of a method of the invention previously described, the verification device compares the result (g^y) supplied by the mathematical function applied to the authentication value V either to the value x or to the product (xp) of the value x and the public key p of the chip corresponding to its secret key s, as a function of the value of the parameter t, where y is equal to the authentication value V and p is the public key of the chip corresponding to its secret key s, as defined by the function $p = g^s$.

The means typically comprise a computer.